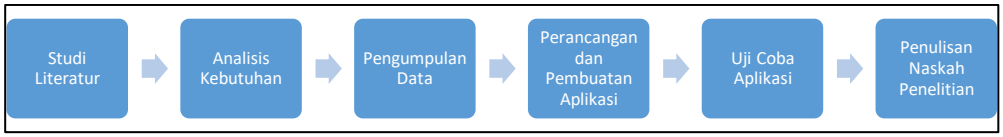
BAB III

METODOLOGI PENELITIAN DAN PERANCANGAN SISTEM

3.1 Metodologi Penelitian

Tahapan metode yang dilaksanakan untuk melakukan penelitian ini adalah sebagai berikut.



Gambar 3.1 Metodologi penelitian

1. Studi Literatur

   Studi literatur dilakukan sebagai pembelajaran terhadap berbagai teori yang memiliki relevansi terhadap perancangan dan pembuatan aplikasi. Teori yang dimaksud antara lain Word2Vec, FastText, t-Distributed Stochastic Neighbor Embedding (t-SNE), Euclidean Distance, dan Cosine Similarity.

2. Analisis Kebutuhan

   Analisis kebutuhan dilakukan untuk menganalisis metode, data, dan proses apa saja yang akan digunakan, penentuan perangkat lunak dan perangkat keras, dan bentuk keluaran apa yang akan dihasilkan oleh aplikasi. Salah satu metode yang digunakan adalah metode untuk mengukur jarak, yang menggunakan *Euclidean Distance* dan *Cosine Similarity*. Digunakan dua teknik pengukuran jarak karena, pada saat vektor pada dimensi tinggi, bukan jarak antar data lagi yang penting, tapi orientasinya atau arahnya dan Cosine Similarity ini digunakan untuk menghitung *similarity* antar data berdasarkan orientasinya. Lalu untuk dimensi rendah digunakan Euclidean Distance ini karena teknik ini direkomendasikan untuk menghitung jarak

pada dimensi rendah, dan ada penelitian lain yang tidak menyarankan Euclidean Distance tidak digunakan untuk menghitung jarak pada dimensi tinggi.

3. Pengumpulan Data

   Pengumpulan data dilakukan agar mendapatkan data yang akan digunakan dalam penelitian. Penelitian ini membutuhkan data model yang diambil dari situs FastText dengan cara mengunduh model yang telah disediakan dari situs serta sekumpulan kata yang didapatkan dari KBBI.

4. Perancangan dan Pembuatan Aplikasi

   Tahap perancangan dilakukan dengan membuat rancangan aplikasi yang akan dibangun, meliputi perancangan *flowchart* yang menggambarkan alur kerja dari aplikasi ini. Pembuatan aplikasi Visualisasi FastText Word2Vec Model menggunakan algoritma t-Distributed Stochastic Neighbor Embedding menggunakan bahasa Python yang akan meng-*output*-kan sebuah *file* yang digunakan untuk visualisasi dan bahasa C# untuk memvisualisasikan data berdasarkan model FastText.

5. Uji Coba Aplikasi

   Proses uji coba aplikasi diperlukan agar dapat menguji aplikasi yang telah dibuat serta memperbaiki jika ada kesalahan di dalamnya. Pengujian akan dilakukan terhadap algoritma t-Distributed Stochastic Neighbor Embedding, serta akan dilakukan evaluasi tingkat akurasi model Word2Vec FastText dalam memperbaiki kesalahan penulisan kata.

6. Penulisan Naskah Penelitian

Penulisan naskah penelitian dilakukan sebagai dokumentasi penelitian sehingga dapat menjadi sarana ilmu pengetahuan dan dapat menjadi referensi untuk penelitian selanjutnya.

3.2 Perancangan Aplikasi

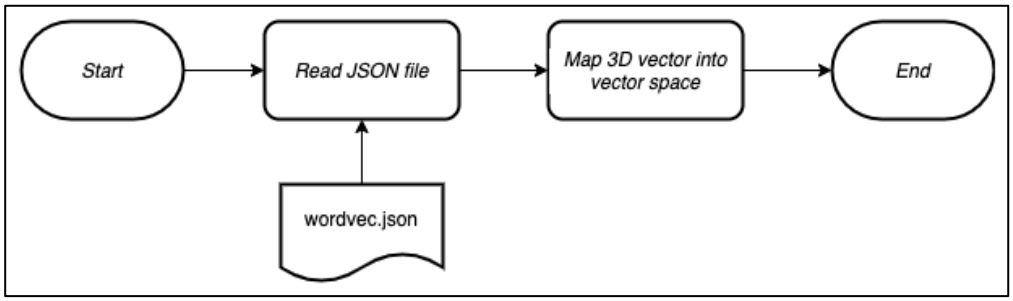
Perancangan aplikasi visualisasi FastText model akan dijabarkan dalam *flowchart* dan rancangan antarmuka berikut ini.

3.2.1 Flowchart

Berikut merupakan *flowchart* dari aplikasi visualisasi, pengukuran tingkat akurasi FastText Word2Vec model, dan pengukuran tingkat akurasi *dimensionality reduction* t-SNE.

A. Flowchart Utama Visualisasi Data

Gambar 3.2 merupakan *flowchart* Visualisasi Data, tahap ini akan memvisualisasikan data JSON yang didapatkan dari sistem. Data JSON yang berisikan setiap kata dan posisi dalam 3 dimensi hasil reduksi dibaca. Kemudian, berdasarkan data tersebut vektor akan dipetakan ke dalam ruang tiga dimensi.

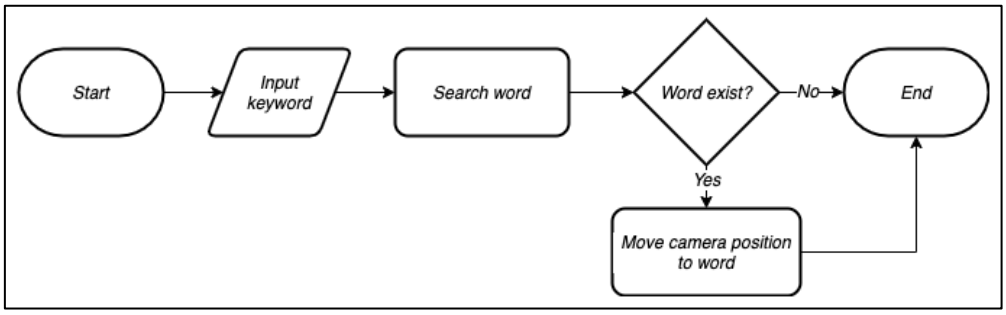


Gambar 3.2 *Flowchart* utama visualisasi data

B. Flowchart *Search Data*

Gambar 3.3 merupakan *flowchart Search Data*, proses ini akan melakukan pencarian kata dalam ruang vektor tiga dimensi berdasarkan kata yang dimasukkan

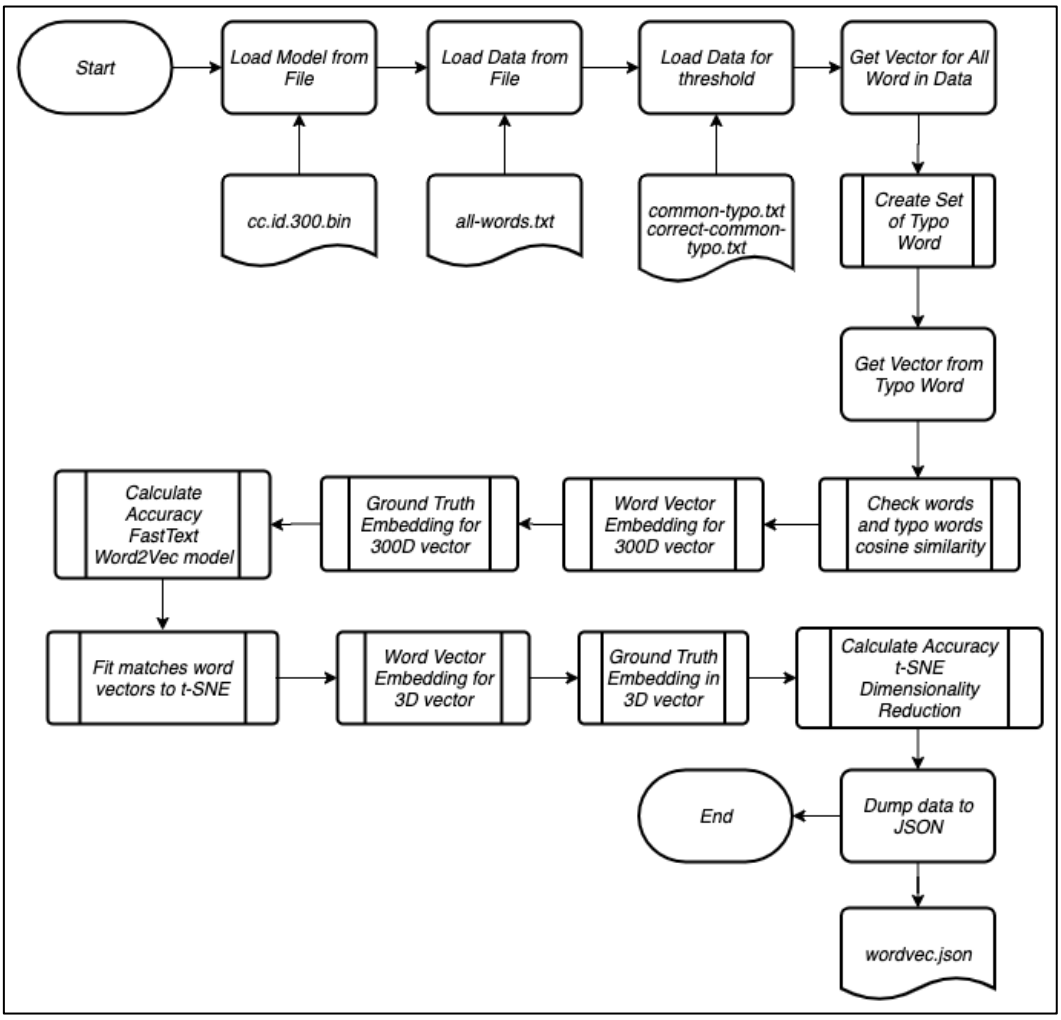
pengguna. Apabila kata ditemukan, maka kamera akan bergerak ke arah posisi kata tersebut, tetapi apabila kata tidak ditemukan, maka kamera tidak berpindah tempat.



Gambar 3.3 *Flowchart search data*

C. Flowchart Utama Sistem

Gambar 3.4 merupakan *flowchart* dari sistem pengukuran tingkat akurasi ini. Awalnya sistem akan membaca model vektor dari *file* cc.id.300.bin yang didapatkan dari situs FastText, *file* tersebut merupakan *pre-trained word vector* Bahasa Indonesia dalam 300 dimensi. Kemudian sistem akan membaca data berisi kumpulan kata yang berimbuhan me- dan di- dari sebuah *file* txt.



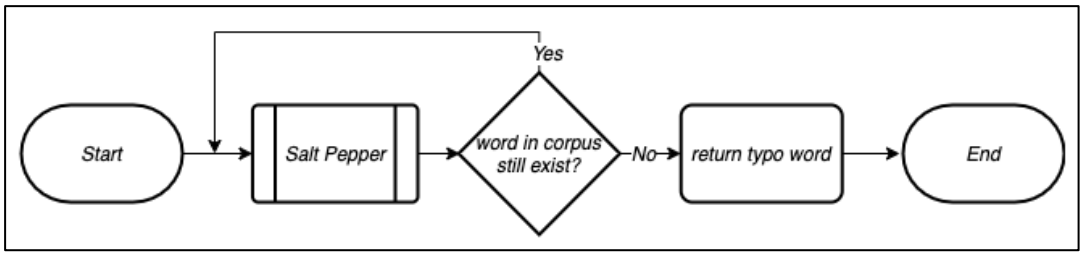
Gambar 3.4 *Flowchart* utama sistem

Lalu sistem akan mendapatkan *word vector* dari kumpulan kata berdasarkan model FastText. Sistem akan membuat kata *typo* dari kumpulan kata tersebut dan akan didapatkan *word vector* dari kumpulan kata *typo*. Selanjutnya kata dengan vektor 300 dimensi dan kata *typo* dengan vektor 300 dimensinya akan digabungkan dan akan dibuat juga *ground truth* yang berisi kata benar. Hasil prediksi FastText diharapkan sesuai dengan *ground truth* yang telah dibuat, semakin banyak prediksi yang sesuai dengan *ground truth* maka semakin besar pula tingkat akurasinya. Selanjutnya akan dilakukan pengukuran akurasi FastText Word2Vec model dalam memprediksi kata benar dari kata yang *typo*. Dari hasil perhitungan tersebut, akan

didapatkan pasangan kata yang diprediksi secara benar, dan vektor dari model pasangan kata yang diprediksi benar akan diproses menggunakan algoritma t-SNE *dimensionality reduction* untuk mendapatkan vektor 3 dimensi yang akan digunakan untuk visualisasi data dan kalkulasi tingkat akurasi t-SNE. Hasil vektor *dimensionality reduction* digabungkan dengan seluruh kata benar dan kata *typo*, juga dibuat *ground truth* yang berisi kata benar yang memiliki vektor 3 dimensi. Lalu akan dilakukan pengukuran tingkat akurasi *dimensionality reduction* yang dilakukan t-SNE, pengukuran akurasi dilakukan dengan menggunakan *euclidean distance*. Terakhir, sistem akan mengeluarkan *file* JSON yang akan digunakan untuk visualisasi data.

D. Flowchart *Create Set of Typo Word*

Gambar 3.5 merupakan *flowchart Create Typo Word*, seluruh kata benar akan di proses melalui fungsi *Salt Pepper*, yang akan mengembalikan kata yang telah di-*typo*-kan.

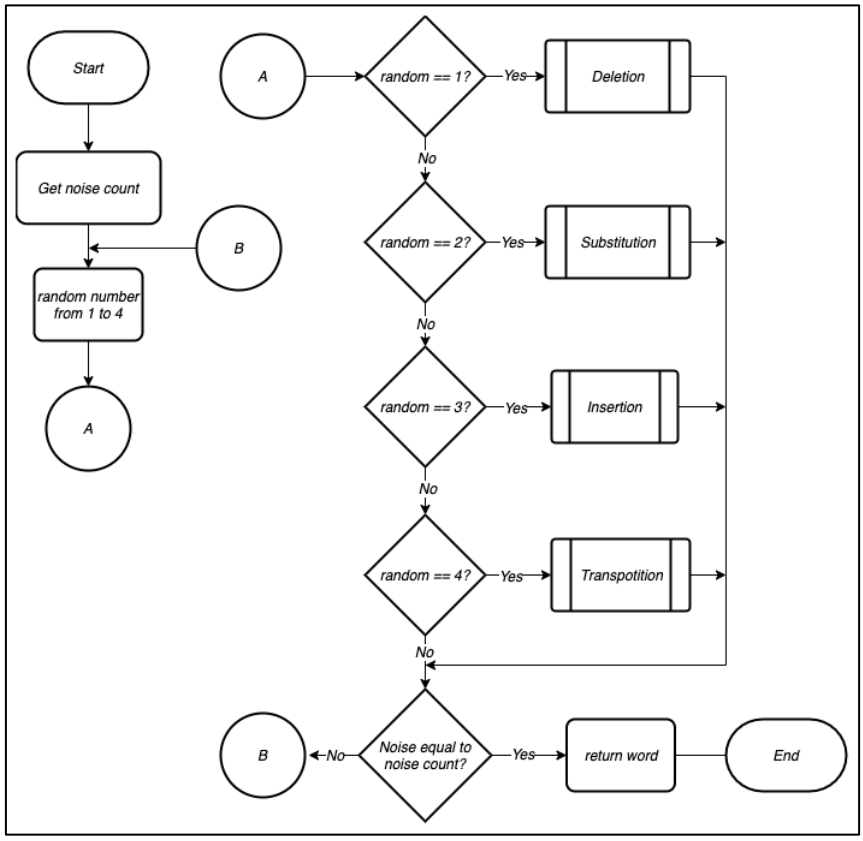


Gambar 3.5 *Flowchart create set of typo word*

E. Flowchart *Salt Pepper*

Gambar 3.6 merupakan flowchart *Salt Pepper*, untuk membuat kata *typo* dibagi menjadi 4 fungsi yang akan dipilih secara acak. Fungsi akan berjalan sebanyak jumlah noise yang diinginkan. Fungsi untuk membuat kata *typo* tersebut adalah *Deletion*, *Substitution*, *Insertion*, dan *Transposition*. Pada penelitian ini juga dilakukan pembuatan kata *typo* yang hanya memiliki satu bentuk kata *typo* saja,

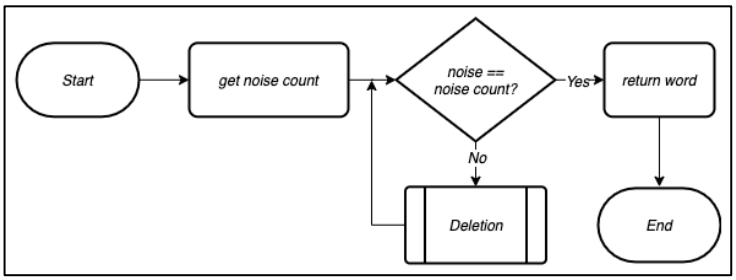
seperti hanya berbentuk *deletion*, hanya berbentuk *insertion*, dan lain-lain. Perubahan bentuk kata *typo* ini dilakukan secara manual pada setiap iterasi penelitian.



Gambar 3.6 *Flowchart salt pepper*

C.1 Flowchart *Salt Pepper Deletion*

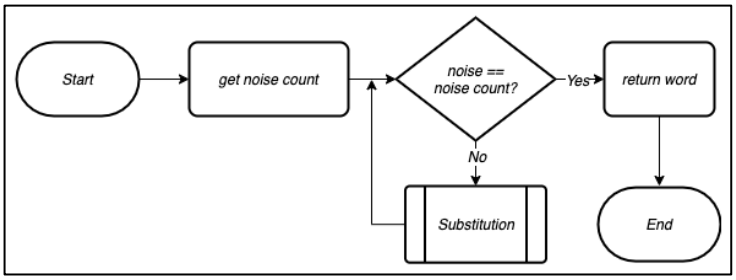
Gambar 3.7 merupakan *flowchart Salt Pepper Deletion*, sebenarnya fungsi ini hampir sama dengan fungsi *Salt Pepper*, hanya saja pada fungsi ini hanya memanggil fungsi *Deletion* untuk membuat kata *typo*. Sehingga kumpulan kata *typo* hanya memiliki tipe *deletion*.



Gambar 3.7 *Flowchart salt pepper deletion*

C.2 Flowchart *Salt Pepper Substitution*

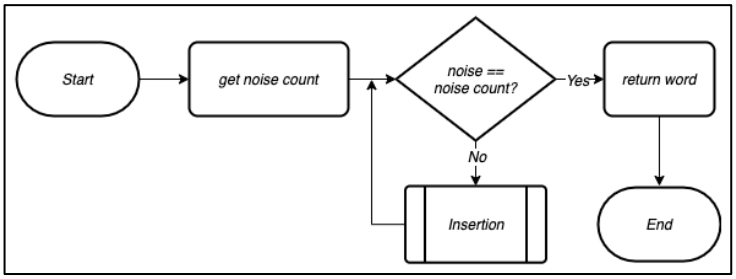
Sama seperti fungsi sebelumnya, Gambar 3.8 merupakan *flowchart Salt Pepper Substitution*, yang hanya memanggil fungsi *Substitution* untuk pembuatan kata *typo*.



Gambar 3.8 *Flowchart salt pepper substitution*

C.3 Flowchart *Salt Pepper Insertion*

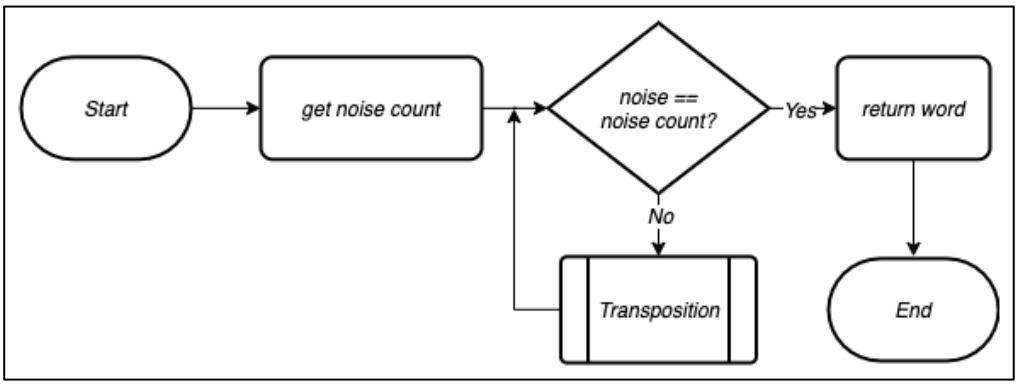
Gambar 3.9 merupakan *flowchart Salt Pepper Insertion* yang hanya memanggil fungsi *Insertion* untuk pembuatan kata *typo*.



Gambar 3.9 *Flowchart salt pepper insertion*

C.4 Flowchart *Salt Pepper Transposition*

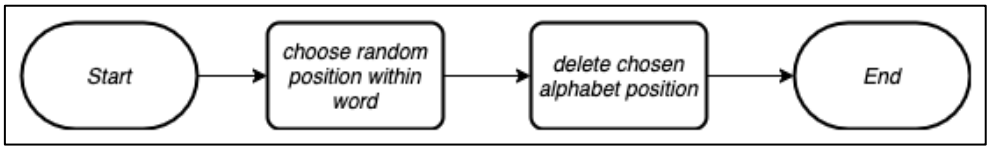
Tidak berbeda dengan fungsi *Salt Pepper* lainnya, Gambar 3.10 merupakan *flowchart Salt Pepper Transposition*, yang hanya memanggil fungsi *Transposition* untuk pembuatan kata *typo*.



Gambar 3.10 *Flowchart salt pepper transposition*

F. Flowchart *Deletion*

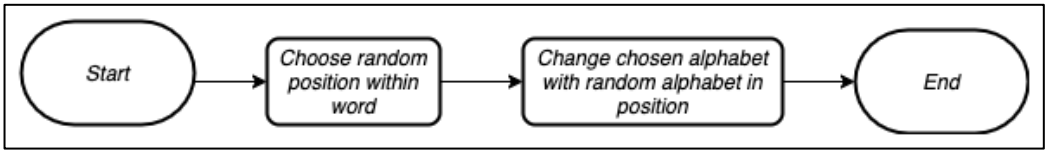
Gambar 3.11 merupakan *flowchart Deletion*, fungsi ini akan menghapus salah satu huruf yang terdapat dalam kata secara acak. Akan dicari satu letak alfabet dalam kata secara acak, lalu alfabet pada posisi tersebut dihapus.



Gambar 3.11 *Flowchart deletion*

G. Flowchart *Substitution*

Gambar 3.12 merupakan *flowchart Substitution*, fungsi ini akan mengubah salah satu huruf yang terdapat dalam kata secara acak. Akan dicari satu letak alfabet dalam kata secara acak, lalu alfabet pada posisi tersebut diubah dengan huruf yang dipilih secara acak.

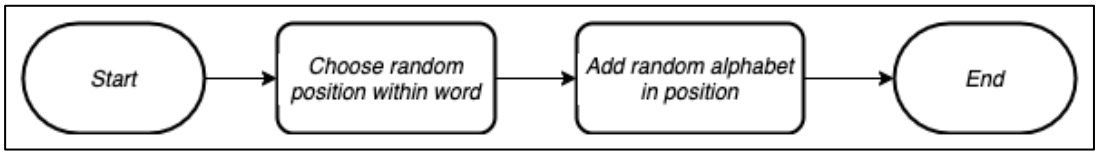


Gambar 3.12 *Flowchart substitution*

H. Flowchart *Insertion*

Gambar 3.13 merupakan *flowchart Insertion*, fungsi ini akan menambahkan satu huruf di samping salah satu huruf yang telah dipilih secara acak. Akan dipilih

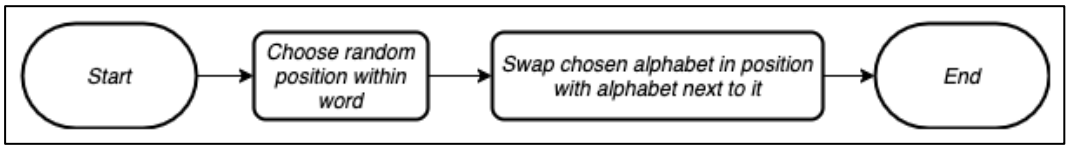
satu letak alfabet dalam kata secara acak, lalu ditambahkan satu huruf yang dipilih secara acak dalam kata pada posisi yang telah dipilih sebelumnya.



Gambar 3.13 *Flowchart insertion*

I. Flowchart *Transposition*

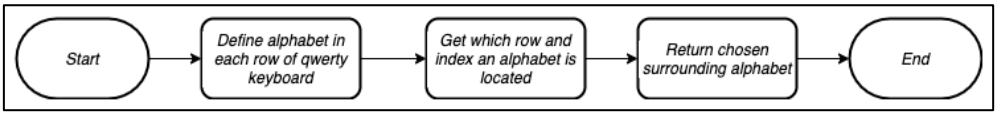
Gambar 3.14 merupakan *flowchart Transposition*, fungsi ini akan menukar dua huruf yang bersebelahan dalam kata yang akan dipilih secara acak. Akan dipilih satu letak alfabet dalam kata secara acak, lalu huruf tersebut ditukar dengan huruf yang bersebelahan.



Gambar 3.14 *Flowchart transposition*

J. Flowchart *Get Keys Around*

Gambar 3.15 merupakan *flowchart Get Keys Around*, fungsi ini akan mengembalikan huruf yang berada di sekeliling huruf yang terpilih secara acak. Awalnya akan dicari di mana huruf tersebut terletak dan akan didapatkan huruf di sekelilingnya.

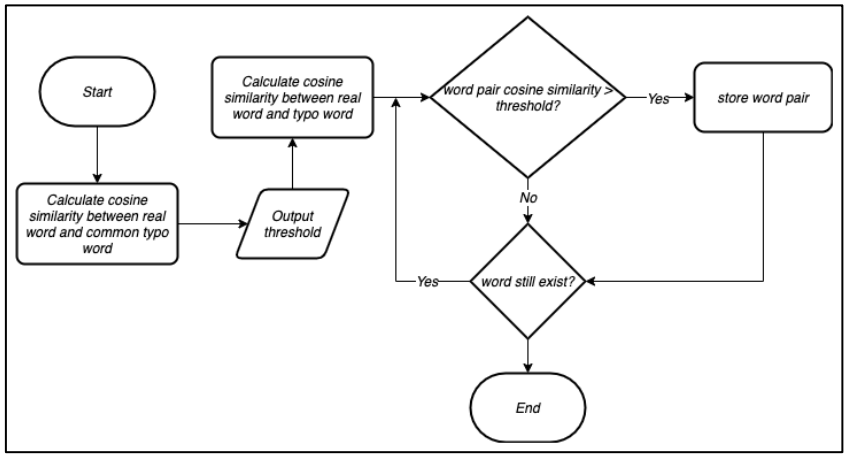


Gambar 3.15 *Flowchart get keys around*

K. Flowchart *Check Words and Typo Words Cosine Similarity*

Gambar 3.16 merupakan *flowchart Check Words and Typo Words Cosine Similarity*, fungsi ini akan menghitung semua *cosine similarity* dari kata asli dengan

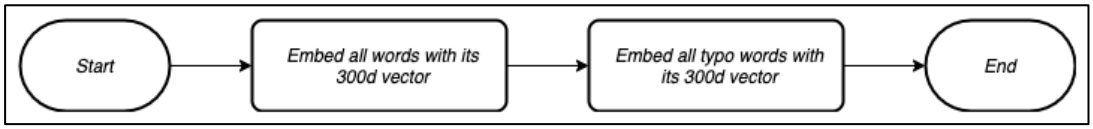
pasangan kata *typo*-nya. Lalu akan dicek nilai *cosine similarity* yang dimiliki, dan kata *typo* yang digunakan adalah yang memiliki nilai *cosine similarity* yang diatas atau sama dengan nilai *threshold*.



Gambar 3.16 *Flowchart check words and typo words cosine similarity*

L. Flowchart *Word Vector Embedding for 300D Vector*

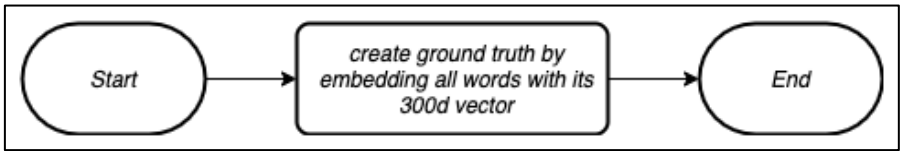
Gambar 3.17 merupakan *flowchart Word Vector Embedding for 300D Vector*, sistem akan menyimpan kata benar bersama pasangan vektor dan vektor kata *typo* juga disimpan bersama pasangan katanya.



Gambar 3.17 *Flowchart word vector embedding for 300D vector*

M. Flowchart *Ground Truth Embedding for 300D Vector*

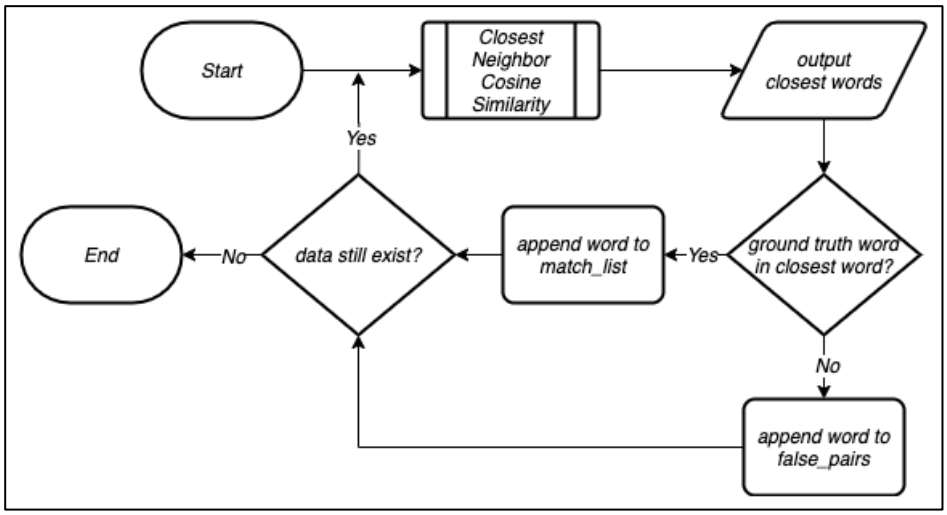
Gambar 3.18 merupakan *flowchart* G*round Truth Embedding for 300D Vector*, pada proses ini akan dilakukan pembuatan *ground truth* yang digunakan sebagai perbandingan apakah hasil prediksi kata *typo* sesuai dengan kata asli atau tidak. *Ground truth* ini didapatkan dari menggabungkan kata benar bersama pasangan vektornya.



Gambar 3.18 *Flowchart ground truth embedding for 300D vector*

N. Flowchart *Calculate FastText Accuracy*

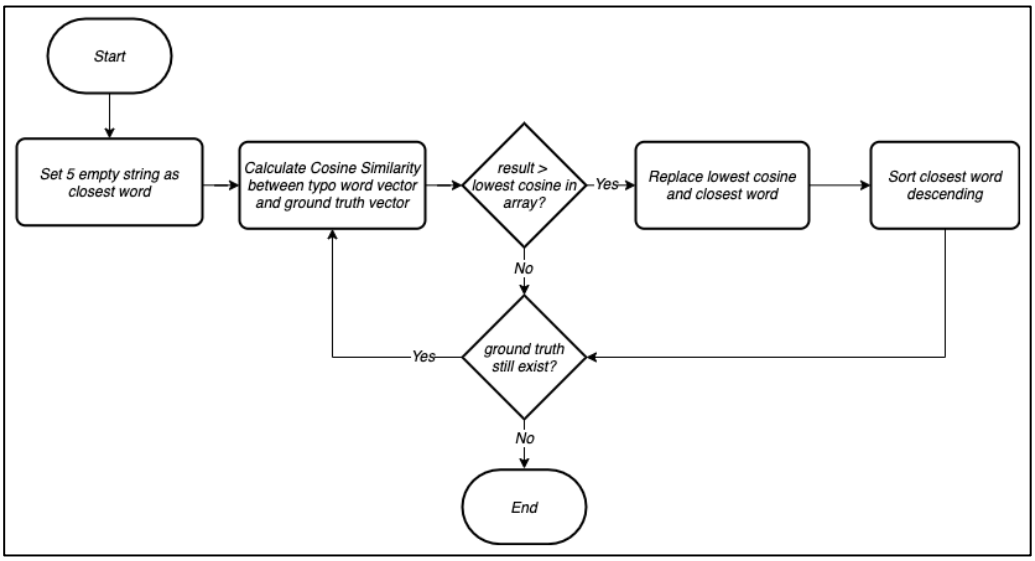
Gambar 3.19 merupakan *flowchart Calculate FastText Accuracy*, proses ini akan mengukur akurasi FastText Word2Vec model dalam memetakan kata *typo* ke bentuk vektor, pengukuran dilakukan dengan *cosine similarity*. Apabila salah satu dari hasil prediksi kata terdekat mengandung kata *ground truth* yang telah dibuat sebelumnya, maka kata tersebut dimasukkan ke dalam *match_list*. Apabila hasil prediksi tidak sesuai, maka kata tersebut dimasukkan ke dalam *false_pairs*.



Gambar 3.19 *Flowchart calculate* FastText *accuracy*

O. Flowchart *Closest Neighbor Cosine Similarity*

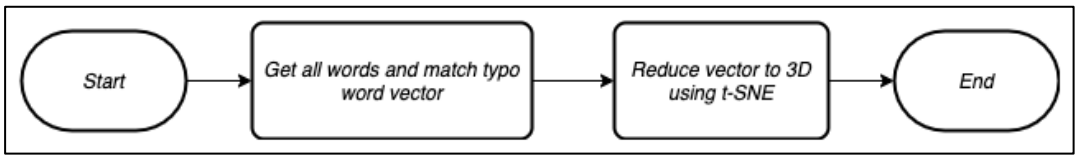
Gambar 3.20 merupakan *flowchart Closest Neighbor Cosine Similarity*, fungsi ini akan mencari 5 kata yang memiliki *cosine similarity* terbesar berdasarkan perbandingan dengan kata *typo*.



Gambar 3.20 *Flowchart closest neighbor cosine similarity*

P. Flowchart *Fit Matches Word Vector to t-SNE*

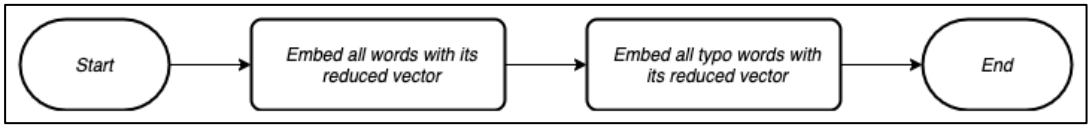
Gambar 3.21 merupakan *flowchart Fit Matches Word Vector to t-SNE*, sistem akan melakukan *dimensionality reduction* untuk vektor kata supaya vektor tersebut dapat divisualisasikan.



Gambar 3.21 *Flowchart fit matches word vector to t-SNE*

Q. Flowchart *Word Vector Embedding for 3D Vector*

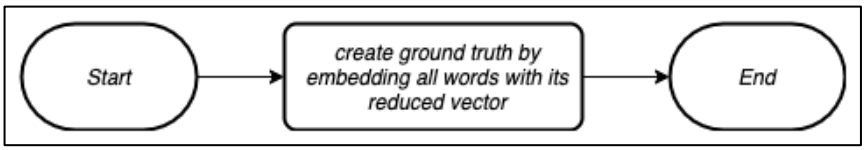
Gambar 3.22 merupakan *flowchart* Word Vector Embedding *for 3D Vector*, sistem akan menyimpan kata benar bersama pasangan vektor hasil *dimensionality reduction* dan hasil *dimensionality reduction* vektor kata *typo* yang diprediksi benar juga disimpan bersama pasangan katanya.



Gambar 3.22 *Flowchart word vector embedding for 3D Vector*

R. Flowchart *Ground Truth Embedding*

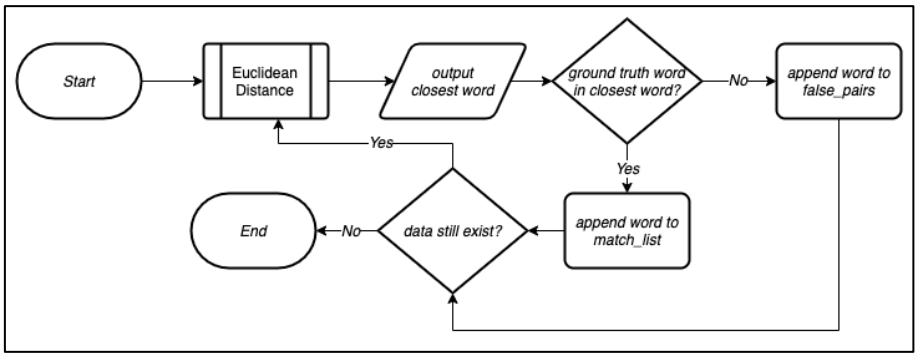
Gambar 3.23 merupakan *flowchart Ground Truth Embedding*, pada proses ini akan dilakukan pembuatan *ground truth* untuk vektor tiga dimensi. *Ground truth* ini didapatkan dari menggabungkan kata benar bersama pasangan vektor hasil *dimensionality reduction*.



Gambar 3.23 *Flowchart ground truth embedding*

S. Flowchart *Calculate Accuracy t-SNE Dimensionality Reduction*

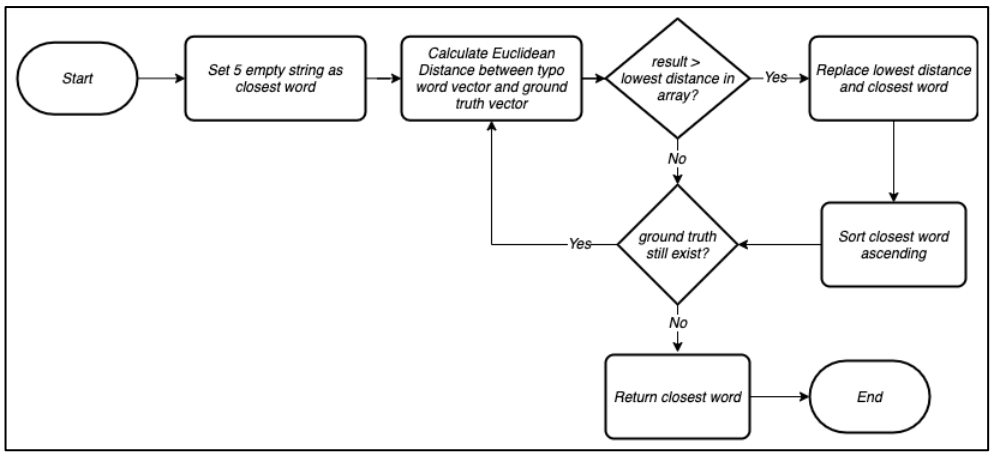
Gambar 3.24 merupakan *flowchart* C*alculate Accuracy t-SNE Dimensionality Reduction*, proses ini akan mengukur akurasi t-SNE dalam menurunkan dimensi vektor, pengukuran dilakukan dengan *euclidean distance*. Apabila hasil prediksi kata mengandung kata *ground truth* yang telah dibuat sebelumnya, maka kata tersebut dimasukkan ke dalam *match_list*. Apabila hasil prediksi tidak sesuai, maka kata tersebut dimasukkan ke dalam *false_pairs*. Semakin banyak kata yang tersimpan pada *match_list*, maka semakin besar pula akurasi t-SNE dalam menurunkan dimensi vektor.



Gambar 3.24 *Flowchart calculate accuracy t-SNE Dimensionality Reduction*

T. Flowchart *Euclidean Distance*

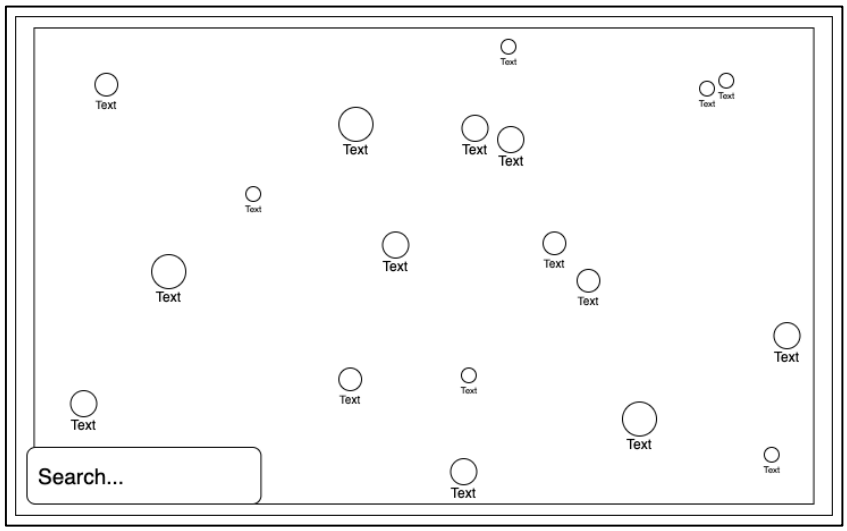
Gambar 3.25 merupakan *flowchart Euclidean Distance*, fungsi ini akan mencari 5 kata yang memiliki jarak terdekat dari kata *typo*. Perhitungan dilakukan untuk semua *ground truth* dibandingkan dengan satu kata *typo*. Apabila hasil perhitungan *distance* lebih kecil daripada jarak terkecil, maka nilai jarak terkecil diubah dengan nilai perhitungan *distance* dan kata pada *ground truth* akan disimpan. Lalu setelah seluruh *ground truth* telah dibandingkan, akan didapatkan kumpulan kata terdekat dari kata *typo* berdasarkan perhitungan *euclidean distance*.



Gambar 3.25 *Flowchart euclidean distance*

3.2.2 Rancangan Antarmuka

Gambar 3.26 merupakan rancangan tampilan halaman visualisasi. Hanya terdapat satu halaman dalam aplikasi Visualisasi FastText Word2Vec model, halaman ini akan menampilkan seluruh kata benar dan kata yang diprediksi secara tepat. Kata tersebut ditampilkan dalam bentuk vektor tiga dimensi dan juga terdapat kolom *search* yang berfungsi untuk mencari kata dalam vektor.



Gambar 3.26 Rancangan tampilan halaman visualisasi 3D *space*